COMSC-210 Lab 9
Associative Arrays

GOOD PROGRAMMING PRACTICES *show / hide*

ABOUT THIS ASSIGNMENT
In this assignment you will write an associative array template and test it. The version of the array implementation that you will write in this lab will be based on chaining, with a static array of STL `lists`. You apply this class to the classic Game Of Life simulation.

After you complete this lab assignment, post the required files to the COMSC server so that you receive the proper credit for this 50-point lab. Your posted files are due at midnight of the evening of the due date indicated in the course outline. Use the "Submit your FA2015 work" link on the class website to post your file for this lab, using the **lab9** folder provided.

LAB 9: Write, Test, and Apply The AssociativeArray Template `[ AssociativeArray.h and AssociativeArrayDriver.cpp ]`
Purpose. In this lab you will implement the public interface outlined in the lecture topic #9 notes.

Requirements. Write **`AssociativeArray.h`** and **`AssociativeArrayDriver.cpp`**. Follow the specification for the public interface in the online lecture notes to write the H file. Use either the dynamic array implementation or the linked structure implementation -- your choice.

The test driver CPP should fully test the template, using the data records from the dvc-schedule.txt file from lab 5a. Open the file, read a hundred or so records and store them, and close the file. Then re-open and re-read those same lines from the file, and use each in a retrieval operation to confirm that it got stored. As you do, remove the matching records and confirm that it's now gone.

Note -- the dvc-schedule.txt file may contain duplicate entries! So test the size of the array vs the number of insertions made, to find out how many duplicates there were. Expect that many duplicates in your testing.

Create a `struct` in the driver for the "key", storing the term and section. Write an `operator==` function that compares the term and section -- it can be a stand-alone or a member, your choice. Something like this:

```
struct TermSection
{
  string term;
  string section;
};

bool operator==(const TermSection& x, const TermSection& y)
{
  if terms don't match, return false
  if sections don't match, return false
  return true
}
```

Use the course name for the "value" -- no need to make this more complicated with a struct for course, name, days, time, room, and name. So you'll write something like this:

```
AssociativeArray<string, TermSection> a;
...
TermSection ts = {term, section};
a[ts] = course;
```

Remember that driver testing needs to include ifndef, object copy, and object assignment testing, as well as using asserts and cout's of expected values in the function testing.

The Classic Game Of Life Simulation

Once the template is tested, apply it to the classic **Game Of Life Simulation**. Click here for the rules to the game of life. Click here for an animation of the game of life. In that page, click the Enjoy Life button to launch an applet. Set ZOOM and SPEED as you like. Right-click here to download a Game Of Life simulation that should work with your **AssociativeArray.h**.

Game Of Life Program I/O. Input: a sequence of pairs of integers, terminated with a pair of -1's. Each pair represents a row and column, and can be any positive or negative integer or zero (but not -1 and -1 together). Remember to NOT read numerics directly from `cin` -- read them as strings and convert. Output: A sequence of X patterns, per the provided code.

Sample. Here's an interesting pattern to run:

1 1 2 2 3 3 3 4 3 5 -1 -1

The above should reach a "steady state" like this after a few generations:

```
 X
X X
X  X
 XX
```

Here's an interesting pattern to run:

1 1 2 2 3 3 4 4 3 4 4 3 5 2 1 4 -1 -1

The 24th generation should look like this:

```
    XXX     XXX
  XXXXX     XXXXX
  XX    X X    XX
 XXXX  XXXXX  XXXX
       X X X
X X    X X X    X X
 XXX X       X XXX
     XXX X XXX
      X     X
```

```
X  X

XXX
```

Note that the 24th generation is symmetric. Because of the rules for the Game Of Life, one should expect that ALL generations beyond the 24th will also be symmetric. Make sure that you test this *well beyond* the 24th generation to convince yourself that your associative array template works right.

How to pause a console program: *show / hide*

GRADING RUBRIC *show/hide*
